

Rule-Based Modeling of Bio-Chemical Networks

Workshop on Modelling in Biology and Medicine – MBM2019

Sandro Stucki

Computer Science Engineering (CSE)
Gothenburg University | Chalmers

Gothenburg, 9 May 2019



UNIVERSITY OF
GOTHENBURG

Why use programming or modeling languages?

Why?

Why use programming or modeling languages?

Syntax

- formal, standardized knowledge representation,
- shareable,
- executable.

Why?

Why use programming or modeling languages?

Syntax

- formal, standardized knowledge representation,
- shareable,
- executable.

Formal semantics

- precise mathematical meaning of programs/models,
- enables formal reasoning.

Why?

Why use programming or modeling languages?

Syntax

- formal, standardized knowledge representation,
- shareable,
- executable.

Formal semantics

- precise mathematical meaning of programs/models,
- enables formal reasoning.

Tooling

- execution, simulation,
- translation, transformation, reduction,
- analysis, verification.

How?

Formal modeling languages – my wish list:

How?

Formal modeling languages – my wish list:

- Simple yet expressive syntax/formalism.

How?

Formal modeling languages – my wish list:

- Simple yet expressive syntax/formalism.
- Formal semantics.

How?

Formal modeling languages – my wish list:

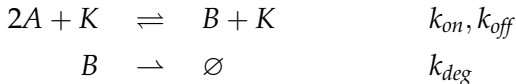
- Simple yet expressive syntax/formalism.
- Formal semantics.
- Automation and tooling for manipulating models.

How?

Formal modeling languages – my wish list:

- Simple yet expressive syntax/formalism.
- Formal semantics.
- Automation and tooling for manipulating models.

Example: Chemical Reaction Networks (CRNs)

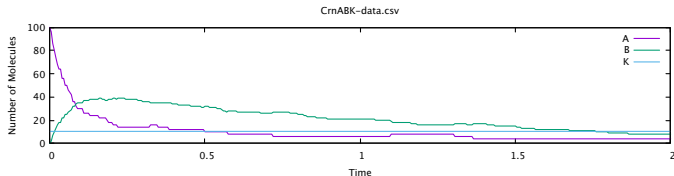
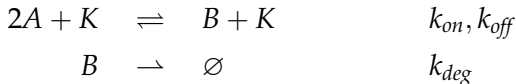


How?

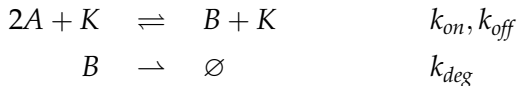
Formal modeling languages – my wish list:

- Simple yet expressive syntax/formalism.
- Formal semantics.
- Automation and tooling for manipulating models.

Example: Chemical Reaction Networks (CRNs)



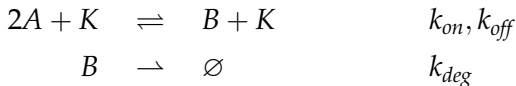
Chemical Reaction Networks (CRNs)



Syntax

- many formats, graphical textual, etc.
- for example, SBML <http://sbml.org/>.

Chemical Reaction Networks (CRNs)



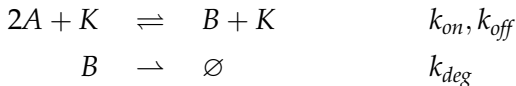
Syntax

- many formats, graphical textual, etc.
- for example, SBML <http://sbml.org/>.

Formal semantics

- stochastic: Markov processes (CTMC),
- differential: rate equations (ODEs),
- others (e.g. Boolean).

Chemical Reaction Networks (CRNs)



Syntax

- many formats, graphical textual, etc.
- for example, SBML <http://sbml.org/>.

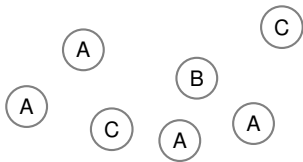
Formal semantics

- stochastic: Markov processes (CTMC),
- differential: rate equations (ODEs),
- others (e.g. Boolean).

Lots of tooling! (See e.g. http://sbml.org/SBML_Software_Guide)

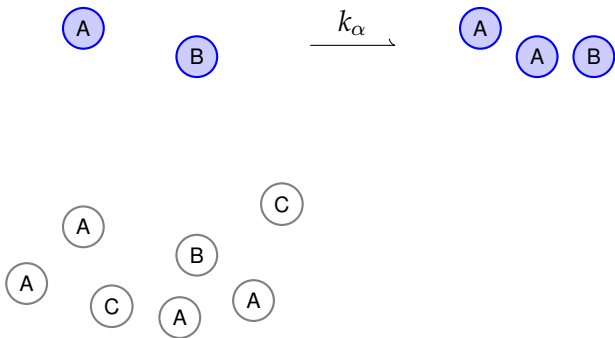
- stochastic simulation (Monte Carlo/Gillespie),
- numerical integration,
- analysis, verification, ...

CRNs as a stochastic process



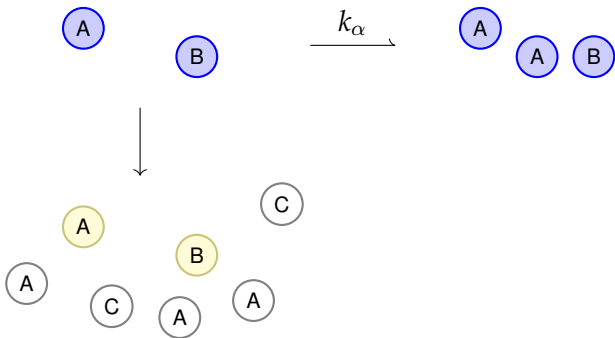
CRNs as a stochastic process

1. Pick a **reaction** α at random (weighted by $k_\alpha \times \# \text{matches}$).



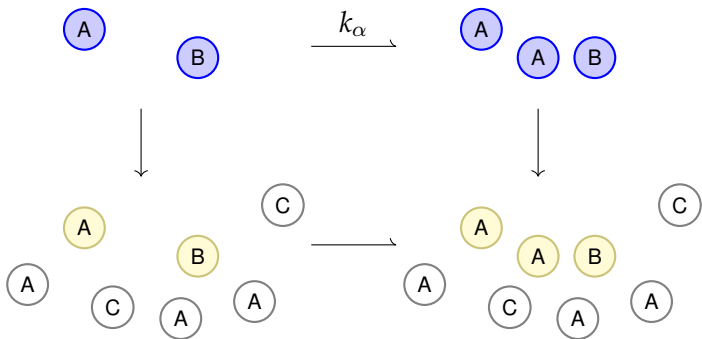
CRNs as a stochastic process

1. Pick a **reaction** α at random (weighted by $k_\alpha \times \# \text{matches}$).
2. Pick a **match** in the **current state** M at random.



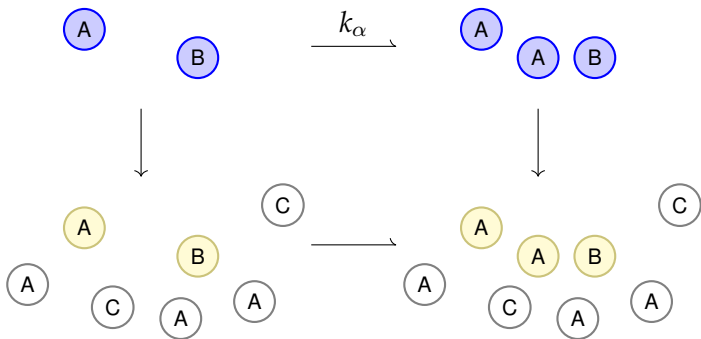
CRNs as a stochastic process

1. Pick a **reaction** α at random (weighted by $k_\alpha \times \# \text{matches}$).
2. Pick a **match** in the **current state** M at random.
3. **Update** M according to α to obtain a **future state** M' .



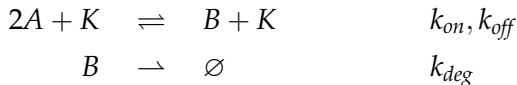
CRNs as a stochastic process

1. Pick a **reaction** α at random (weighted by $k_\alpha \times \# \text{matches}$).
2. Pick a **match** in the **current state** M at random.
3. **Update** M according to α to obtain a **future state** M' .
4. Advance time (Poisson process).



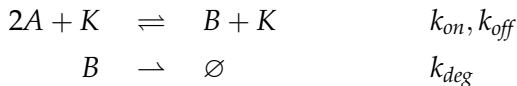
Rate equations (continuous semantics)

- A system of ordinary differential equations (ODEs).
- State space is a vector of concentrations/densities.
- Derivatives are proportional to production/consumption of molecules by rules (mass action).



Rate equations (continuous semantics)

- A system of ordinary differential equations (ODEs).
- State space is a vector of concentrations/densities.
- Derivatives are proportional to production/consumption of molecules by rules (mass action).



$$\frac{d}{dt}[A] = 2k_{off}[B][K] - \frac{1}{2}k_{on}[A]^2[K]$$

$$\frac{d}{dt}[B] = \frac{1}{2}k_{on}[A]^2[K] - 2k_{off}[B][K] - k_{deg}[B]$$

Rate equations (continuous semantics)

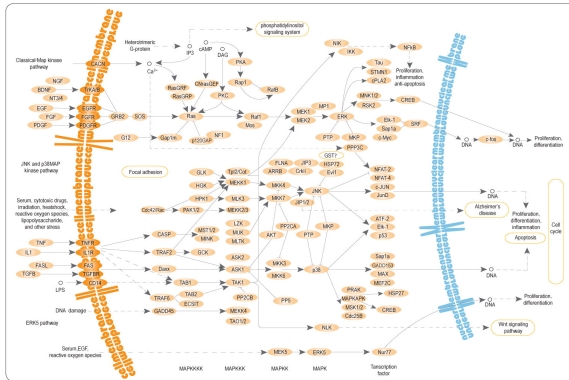
- A system of ordinary differential equations (ODEs).
- State space is a vector of concentrations/densities.
- Derivatives are proportional to production/consumption of molecules by rules (mass action).

$$\begin{aligned}\frac{d}{dt}[A] &= 2k_{off}[B][K] - \frac{1}{2}k_{on}[A]^2[K] \\ \frac{d}{dt}[B] &= \frac{1}{2}k_{on}[A]^2[K] - 2k_{off}[B][K] - k_{deg}[B]\end{aligned}$$

- The REs approximate the behavior of the CTMC semantics in the limit of large molecule counts (abstraction).
- Can be solved by numerical integration (more efficient than stochastic simulation for large systems).

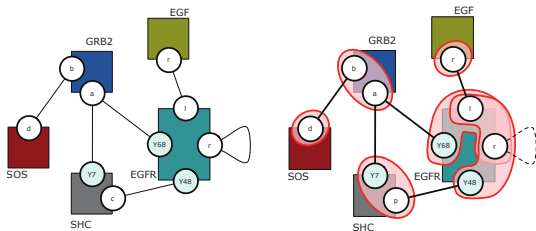
The challenge

Combinatorial Explosion



MAPK pathway, diagram by Kosgrim (Wikipedia), 2007.

Biochemical reaction networks

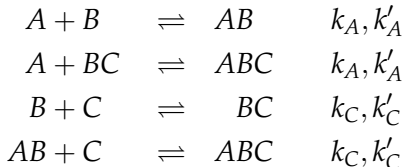


Maps for the early EGF model, Figure 5, [Danos et al., 2010].

Biochemical reaction networks can suffer from high combinatorial complexity:

- Molecules interact through domains.
- A single chemical species may display multiple domains.
- The number of species exhibits a **combinatorial explosion** w.r.t. the number of domain interactions.

Biochemical reaction networks

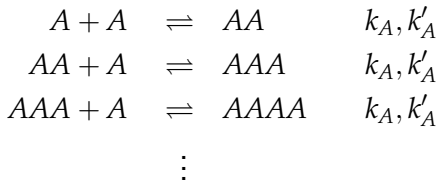


Biochemical reaction networks can suffer from high combinatorial complexity:

- Molecules interact through domains.
- A single chemical species may display multiple domains.
- The number of species exhibits a **combinatorial explosion** w.r.t. the number of domain interactions.

Polymers

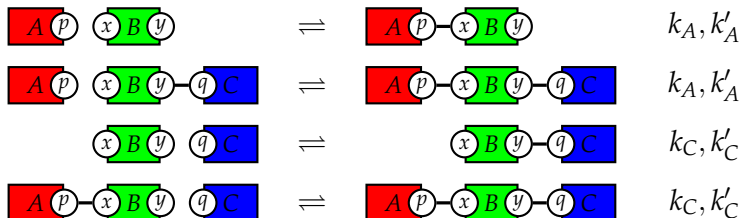
Worst case: polymerization reactions involve an **infinite** number of species/reactions.



Rule-Based Models (RBMs)

Rule-based modeling languages such as Kappa and BNGL¹ have been introduced to deal with this combinatorial complexity.

- **Rules** describe interaction on the domain level.
- A single rule captures a (possibly infinite) **set of reactions**.

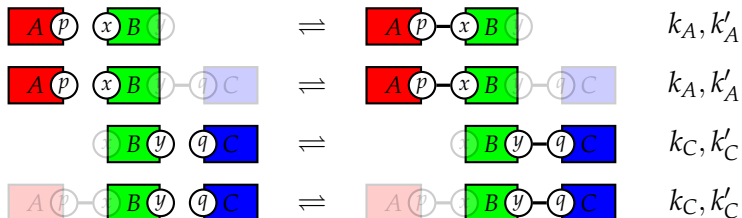


¹See e.g. [Danos et al., 2007] and [Blinov et al., 2004]

Rule-Based Models (RBMs)

Rule-based modeling languages such as Kappa and BNGL¹ have been introduced to deal with this combinatorial complexity.

- **Rules** describe interaction on the domain level.
- A single rule captures a (possibly infinite) **set of reactions**.

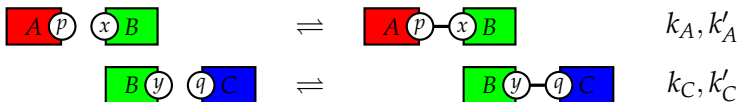


¹See e.g. [Danos et al., 2007] and [Blinov et al., 2004]

Rule-Based Models (RBMs)

Rule-based modeling languages such as Kappa and BNGL¹ have been introduced to deal with this combinatorial complexity.

- **Rules** describe interaction on the domain level.
- A single rule captures a (possibly infinite) **set of reactions**.

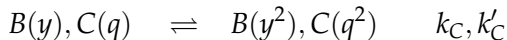
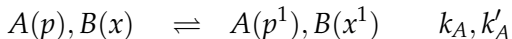
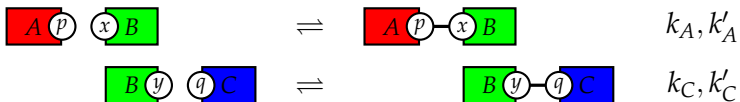


¹See e.g. [Danos et al., 2007] and [Blinov et al., 2004]

Rule-Based Models (RBMs)

Rule-based modeling languages such as Kappa and BNGL¹ have been introduced to deal with this combinatorial complexity.

- **Rules** describe interaction on the domain level.
- A single rule captures a (possibly infinite) **set of reactions**.



¹See e.g. [Danos et al., 2007] and [Blinov et al., 2004]

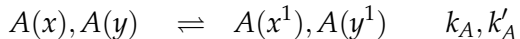
Polymers (cont.)

Polymerization reactions can be expressed compactly.

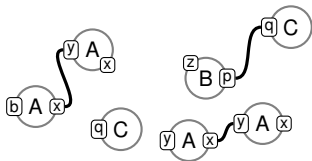


Polymers (cont.)

Polymerization reactions can be expressed compactly.

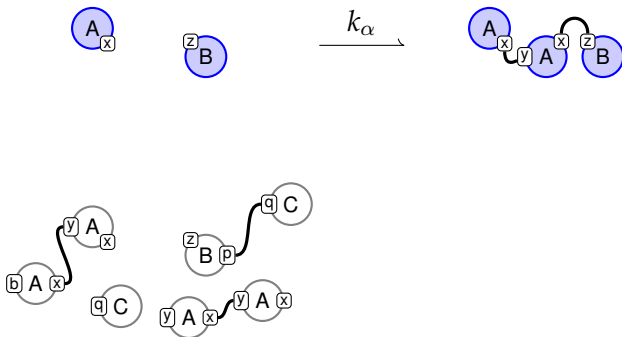


RBM as a stochastic process



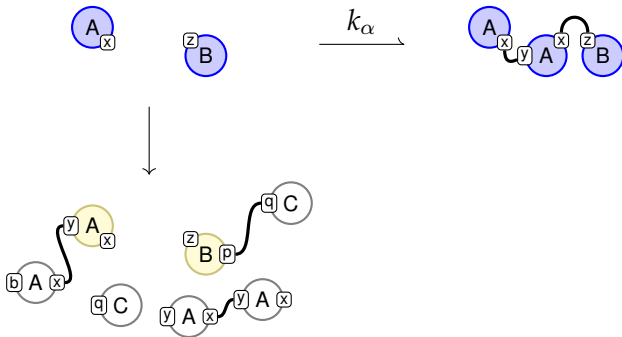
RBM as a stochastic process

1. Pick a rule α at random (weighted by $k_\alpha \times \# \text{matches}$).



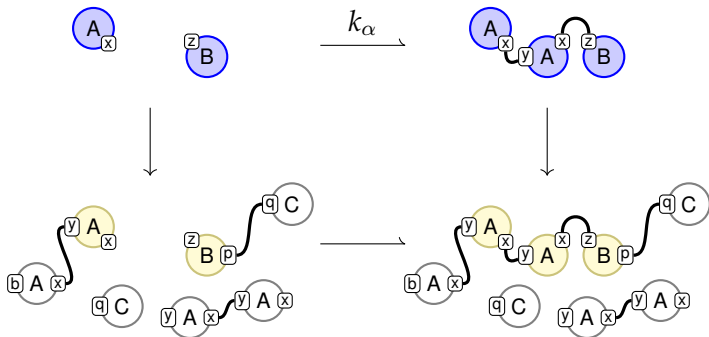
RBM as a stochastic process

1. Pick a **rule** α at random (weighted by $k_\alpha \times \# \text{matches}$).
2. Pick a **match** in the **current state** G at random.



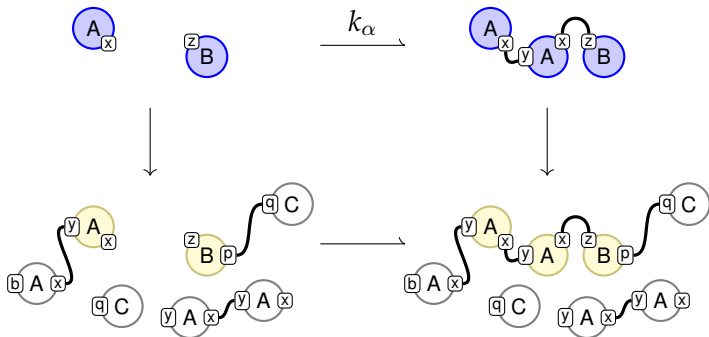
RBM as a stochastic process

1. Pick a **rule** α at random (weighted by $k_\alpha \times \# \text{matches}$).
2. Pick a **match** in the **current state** G at random.
3. **Update** G according to α to obtain a **future state** G' .

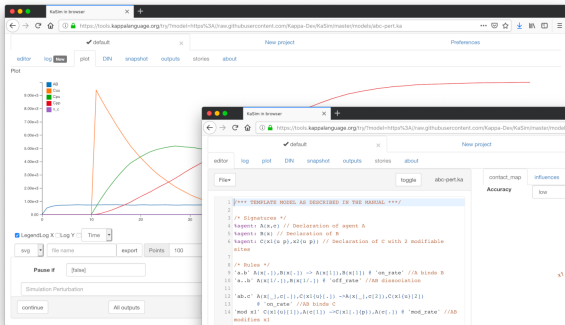


RBM as a stochastic process

1. Pick a **rule** α at random (weighted by $k_\alpha \times \# \text{matches}$).
2. Pick a **match** in the **current state** G at random.
3. **Update** G according to α to obtain a **future state** G' .
4. Advance time (Poisson process).



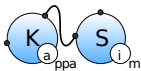
Tooling – Kappa



A screenshot of the Kappa web interface showing the model code and a state transition diagram. The code is as follows:

```
1 |>*** TEMPLATE MODEL AS DESCRIBED IN THE MANUAL ***|>
2 |
3 | /* Elpatoxoa */
4 | agent A(x) // Declaration of agent A
5 | agent B(x) // Declaration of B
6 | agent C(x1(x p),x2(x p)) // Declaration of C with 2 modifiable
  size
7 |
8 | /* Bindings */
9 | "a.b" A(x1(-),B(x1(-)) -> A(x11),B(x11)) # "on_rate" //A binds B
10 | "a..b" A(x1(-),B(x1(-)) # "off_rate" //AB dissociation
11 |
12 | "ab..c" A(x1_1,c(-),C(x1(0)-)) -> A(x1_1,c(2)),C(x1(0)2)
13 | # "on_rate" //AB binds C
14 | "mod x1" C(x1(0)1),A(c(1)) -> C(x11_1(p)),A(c(1)) # "mod_rate" //AB
  modify x1
15 | "a..c" A(x1_1,c(-),C(x1(p)1_1,x21_1(0)) ->
16 | A(x1_1,c(1)),C(x1(p)1_1,x21_1(0)) # "on_rate" //A binds C on x2
17 | "mod x2" A(x1_1,c(1),C(x1(0),x2(0)1) ->
```

The diagram on the right shows a circular state transition graph with nodes 'a', 'b', 'c', 'x1', and 'x2' and directed edges representing transitions. Below the diagram are controls for 'Pause #', 'Period', and 'start' buttons.



<https://kappalanguage.org/>

Rate equations for rule-based models

Stochastic simulation of RBMs is easy (using a variant of Gillespie/Monte Carlo method [Danos et al., 2007]).

Rate equations for rule-based models

Stochastic simulation of RBMs is easy (using a variant of Gillespie/Monte Carlo method [Danos et al., 2007]).

Finding rate equations for RBMs is more tricky...

Rate equations for rule-based models

Stochastic simulation of RBMs is easy (using a variant of Gillespie/Monte Carlo method [Danos et al., 2007]).

Finding rate equations for RBMs is more tricky...

Problem: to find the rate equations of a rule-based model we need to refine it into its ground reactions, possibly at the cost of a combinatorial explosion.

Rate equations for rule-based models

Stochastic simulation of RBMs is easy (using a variant of Gillespie/Monte Carlo method [Danos et al., 2007]).

Finding rate equations for RBMs is more tricky...

Problem: to find the rate equations of a rule-based model we need to refine it into its ground reactions, possibly at the cost of a combinatorial explosion.

$$\begin{aligned}\frac{d}{dt}[A] &= k'_A([AB] + [ABC]) && - k_A[A]([B] + [BC]) \\ \frac{d}{dt}[C] &= k'_C([BC] + [ABC]) && - k_C[C]([B] + [AB]) \\ \frac{d}{dt}[B] &= k'_A[AB] + k'_C[BC] && - [B](k_A[A] + k_C[C]) \\ \frac{d}{dt}[AB] &= k_A[A][B] + k'_C[BC] && - [AB](k'_A + k_C[C]) \\ \frac{d}{dt}[BC] &= k_C[B][C] + k'_A[ABC] && - [BC](k'_C + k_A[A]) \\ \frac{d}{dt}[ABC] &= k_A[A][BC] + k_C[AB][C] && - [ABC](k'_A + k'_C)\end{aligned}$$

Abstracting rate equations

If we allow ourselves to write ODEs over a suitable set of **sub-species**, we can **reduce** the system of rate equations:

$$\begin{array}{r}
 \left. \begin{array}{l}
 \frac{d}{dt}[A] = k'_A([AB] + [ABC]) - k_A[A]([B] + [BC]) \\
 \dots
 \end{array} \right\} 6 \text{ ODEs} \\
 \begin{array}{c} \downarrow \\ \dots \\ \downarrow \end{array} \\
 \left. \begin{array}{l}
 \frac{d}{dt}[A] = \frac{d}{dt}[B?] = k'_A[AB?] - k_A[A][B?] \\
 \frac{d}{dt}[AB?] = k_A[A][B?] - k'_A[AB?]
 \end{array} \right\} 3 \text{ ODEs}
 \end{array}$$

$[B?] = [B] + [BC]$
 $[AB?] = [AB] + [ABC]$

Abstracting rate equations

If we allow ourselves to write ODEs over a suitable set of **sub-species**, we can **reduce** the system of rate equations:

$$\begin{array}{l}
 \left. \begin{array}{l}
 \frac{d}{dt}[A] = k'_A([AB] + [ABC]) - k_A[A]([B] + [BC]) \\
 \dots
 \end{array} \right\} 6 \text{ ODEs} \\
 \begin{array}{c} \downarrow \\ \dots \\ \downarrow \end{array} \\
 \begin{array}{l}
 [B?] = [B] + [BC] \\
 [AB?] = [AB] + [ABC]
 \end{array} \\
 \left. \begin{array}{l}
 \frac{d}{dt}[A] = \frac{d}{dt}[B?] = k'_A[AB?] - k_A[A][B?] \\
 \frac{d}{dt}[AB?] = k_A[A][B?] - k'_A[AB?]
 \end{array} \right\} 3 \text{ ODEs}
 \end{array}$$

$B? = \textcircled{x} \boxed{B}$ and $AB? = \boxed{A} \textcircled{p} \textcircled{x} \boxed{B}$ are called **fragments**

Abstracting rate equations

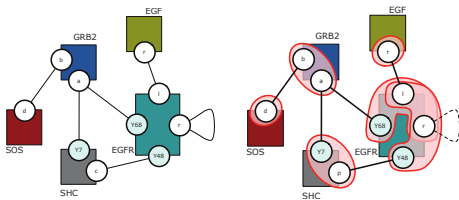
If we allow ourselves to write ODEs over a suitable set of **sub-species**, we can **reduce** the system of rate equations:

$$\begin{array}{r}
 \left. \begin{array}{l}
 \frac{d}{dt}[A] = k'_A([AB] + [ABC]) - k_A[A]([B] + [BC]) \\
 \dots
 \end{array} \right\} 6 \text{ ODEs} \\
 \begin{array}{c} \downarrow \\ \dots \\ \downarrow \end{array} \\
 \left. \begin{array}{l}
 \frac{d}{dt}[A] = \frac{d}{dt}[B?] = k'_A[AB?] - k_A[A][B?] \\
 \frac{d}{dt}[AB?] = k_A[A][B?] - k'_A[AB?]
 \end{array} \right\} 3 \text{ ODEs}
 \end{array}$$

$$\begin{array}{c}
 [B?] = [B] + [BC] \\
 [AB?] = [AB] + [ABC]
 \end{array}$$

$B? = \textcircled{x} \boxed{B}$ and $AB? = \boxed{A} \textcircled{p} \textcircled{x} \boxed{B}$ are called **fragments**

Automated model reduction



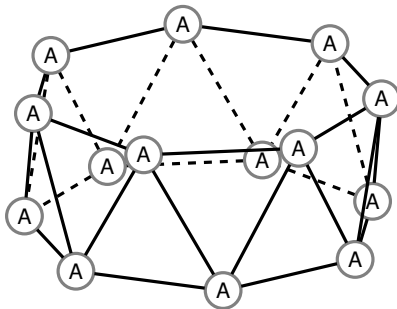
Maps for the early EGF model, Figure 5, [Danos et al., 2010].

Automated extraction of fragments using model analysis:

- reduction of rate equations [Danos et al., 2010],
- reduction of CTMCs [Ferret et al., 2012].

Self assembly

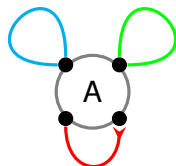
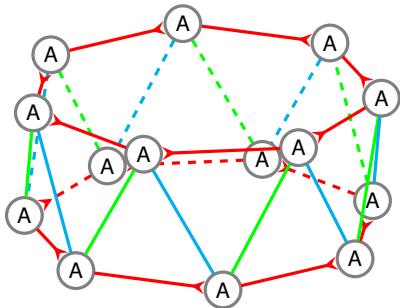
What if we want to study the self-assembly of biochemical complexes, such as molecular machines or signaling complexes?



A simple Kappa model (agents)

Let's try to write a simple Kappa model for assembling our example complex. We need

- a single type of agent with four sites (representing monomers), and
- three types of links (representing the weak bonds among monomers).

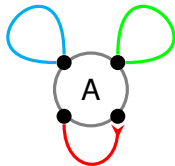


Contact graph

A simple Kappa model (rules)

To assemble bigger and bigger parts of our drum out of a pool of individual agents and smaller parts, we use

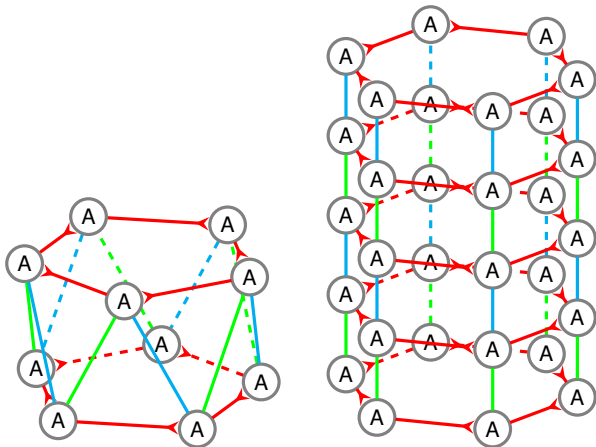
- three atomic association rules, and
- three atomic dissociation rules.



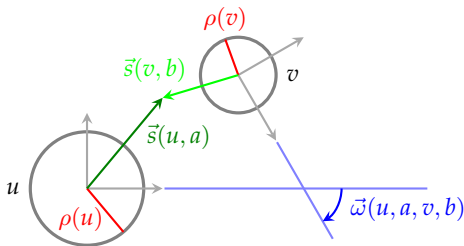
Contact graph

The problem

- Kappa has no notion of **space** or **geometry**.
- Our model describes the self-assembly of **other structures** too (differently sized rings, tubules, etc.)



GeEK – Adding geometry

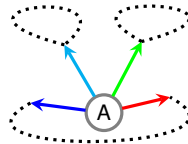
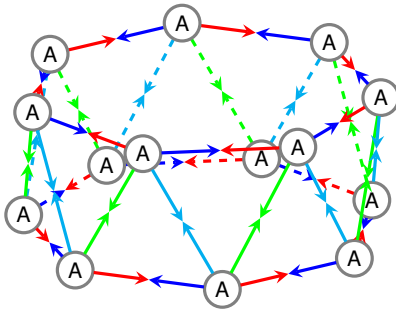


Let's add **three-dimensional geometry** to our site graphs:

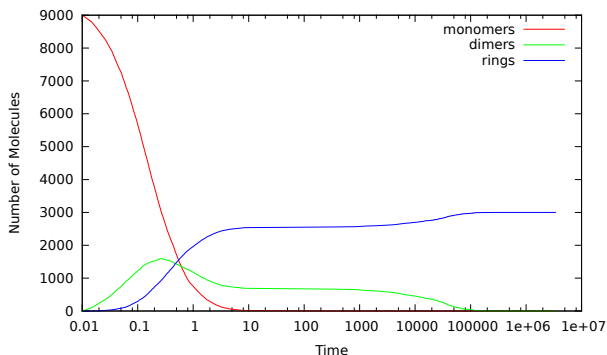
- agents may have **radii**,
- sites may have **positions** (w.r.t. their agents),
- each link may have an **orientations** (constraining the orientations of the bound agents).

A better model

Our simple model extended with geometry:



Case study – self assembly of simple rings



A simple model inspired by [Deeds et al., 2012].

- Homomeric three-membered rings self-assemble from monomers with uniform affinities between binding sites.
- The concentration of rings experiences a **plateau**.

Thank you!

Collaborators

- Vincent Danos, ENS Paris & CNRS
- Tobias Heindel, U.H. Manoa
- Ricardo Honorato-Zimmer, UoE
- Jean Krivine, Paris Diderot & CNRS
- Jérôme Feret, ENS & INRIA Paris
- Russ Harmer, ENS Lyon & CNRS
- Walter Fontana, HMS
- Pierre Bouillier, HMS



More resources

Kappa <https://kappalanguage.org/>

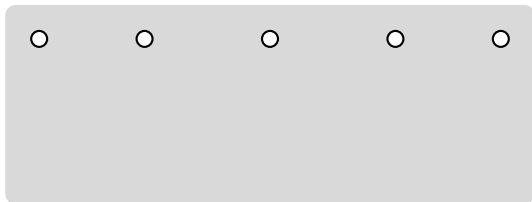
GeEK <https://github.com/sstucki/lms-kappa/>

Additional slides

Modeling more general networks

Modeling more general networks

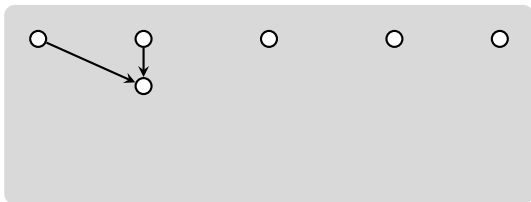
Example: a birth process tracking ancestry.



Genealogy

Modeling more general networks

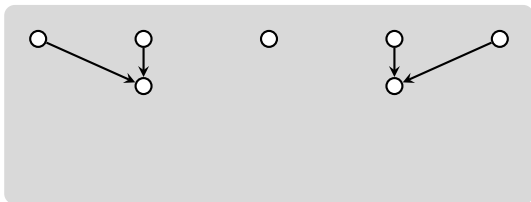
Example: a birth process tracking ancestry.



Genealogy

Modeling more general networks

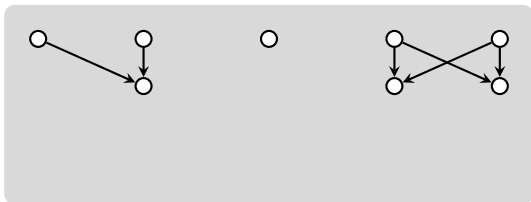
Example: a birth process tracking ancestry.



Genealogy

Modeling more general networks

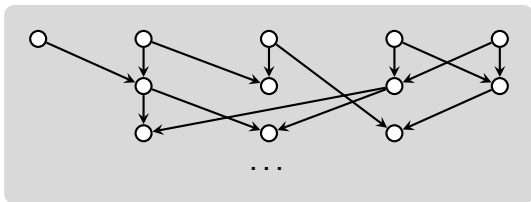
Example: a birth process tracking ancestry.



Genealogy

Modeling more general networks

Example: a birth process tracking ancestry.



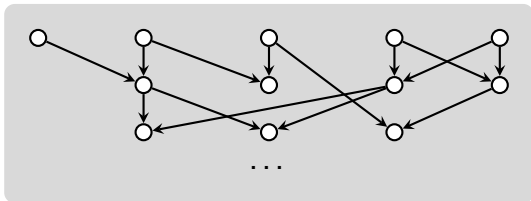
Genealogy

Modeling more general networks

Example: a birth process tracking ancestry.



- Dynamics are given by **graph rewrite rules**.
- Observables are tracked through **graph patterns**.



Genealogy

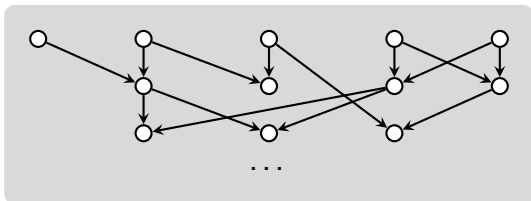
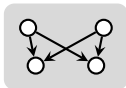
Modeling more general networks

Example: a birth process tracking ancestry.



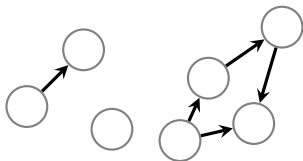
- Dynamics are given by **graph rewrite rules**.
- Observables are tracked through **graph patterns**.

Example: siblings



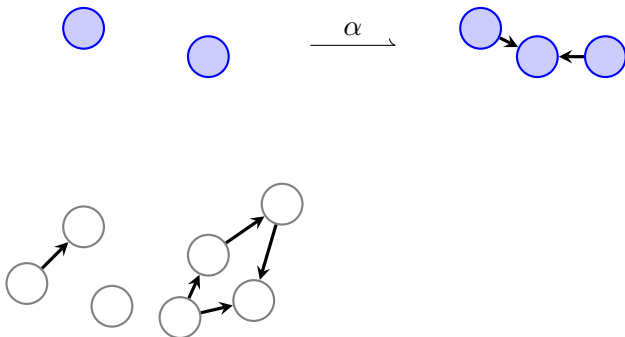
Genealogy

Graph rewriting as a stochastic process



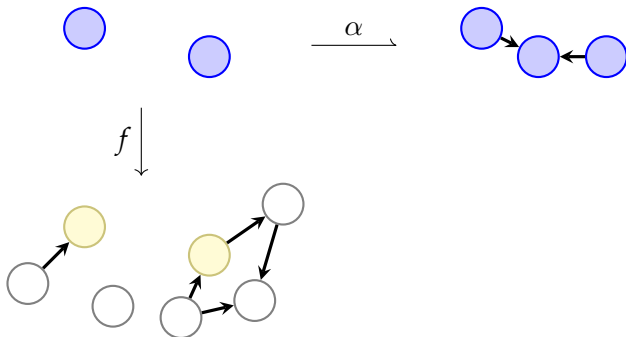
Graph rewriting as a stochastic process

1. Pick a rule α at random (weighted by $k_\alpha \times \# \text{matches}$).



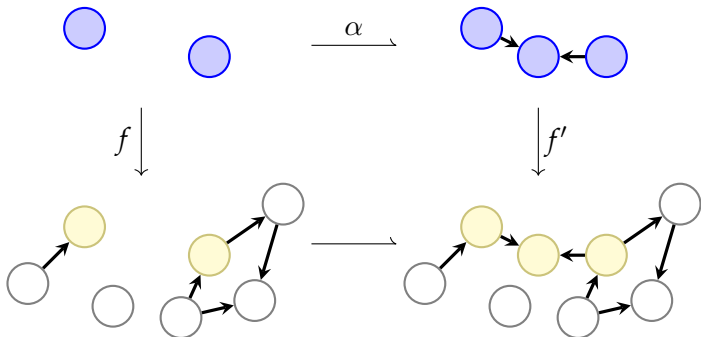
Graph rewriting as a stochastic process

1. Pick a **rule** α at random (weighted by $k_\alpha \times \# \text{matches}$).
2. Pick a **match** in the **current state** G at random.



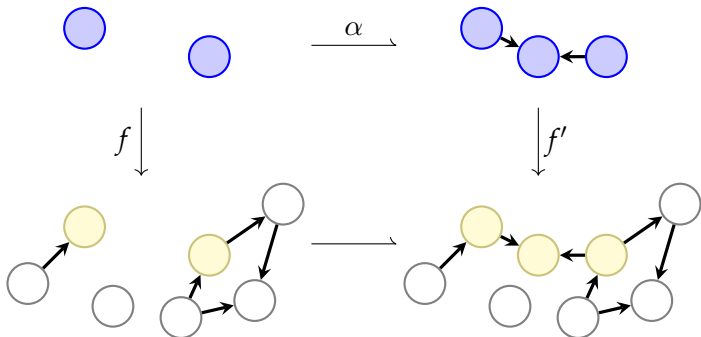
Graph rewriting as a stochastic process

1. Pick a **rule** α at random (weighted by $k_\alpha \times \# \text{matches}$).
2. Pick a **match** in the **current state** G at random.
3. Update G according to α to obtain a **future state** G' .



Graph rewriting as a stochastic process

1. Pick a **rule** α at random (weighted by $k_\alpha \times \# \text{matches}$).
2. Pick a **match** in the **current state** G at random.
3. **Update** G according to α to obtain a **future state** G' .
4. Advance time (Poisson process).



Stochastic graph rewriting

- State space is a **graph-like structure**.
- Transitions induced by **graph rewrite rules**.
- Rates are proportional to number of LHS matches (still mass action).

Stochastic simulation using variants of Gillespie/Monte Carlo is still possible but potentially inefficient (sub-graph isomorphism).

Rate equations for graph-like systems

Problems:

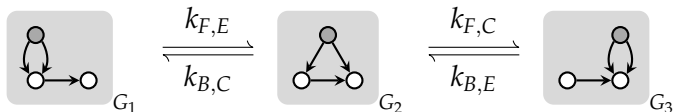
- No more contact map!
- What is a species, a ground refinement?
- Does it even make sense to talk about densities?

The notion of a fragment still makes sense if we switch to a more **algebraic** construction [Danos et al., 2015].

Bonus: the more general approach allows us to also derive ODEs for higher-order moments (variance, skew, etc.)!

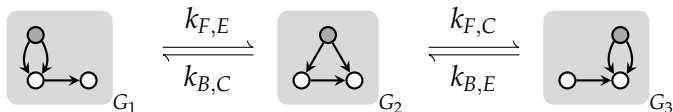
Example: two-legged DNA walker

How fast are two-legged walkers moving along a DNA strand? [Stukalin et al., 2005]



Example: two-legged DNA walker

How fast are two-legged walkers moving along a DNA strand?
[Stukalin et al., 2005]



The combined mean velocity is given by

$$V = \frac{1}{2} (k_{F,E} \mathbb{E}_p([G_1]) + k_{F,C} \mathbb{E}_p([G_2]) - k_{B,E} \mathbb{E}_p([G_3]) - k_{B,C} \mathbb{E}_p([G_2]))$$

Two-legged DNA walker (cont.)

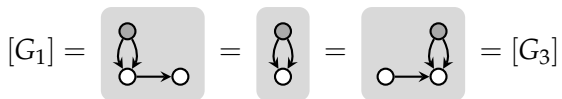
The initial system of ODEs

$$\begin{aligned}
 \frac{d}{dt} \text{[Walker 1]} &= k_{F,E} \text{[Walker 2]} - k_{B,C} \text{[Walker 1]} - k_{F,C} \text{[Walker 1]} + k_{B,E} \text{[Walker 1]} \\
 \frac{d}{dt} \text{[Walker 2]} &= -k_{F,E} \text{[Walker 2]} + k_{B,C} \text{[Walker 1]} + k_{F,C} \text{[Walker 2]} - \dots \\
 \frac{d}{dt} \text{[Walker 3]} &= k_{F,E} \text{[Walker 3]} - k_{B,C} \text{[Walker 3]} - k_{F,C} \text{[Walker 3]} + \dots \\
 \frac{d}{dt} \text{[Walker 4]} &= -k_{F,E} \text{[Walker 4]} + k_{B,C} \text{[Walker 3]} + k_{F,C} \text{[Walker 4]} - \dots \\
 \frac{d}{dt} \text{[Walker 5]} &= \dots
 \end{aligned}$$

The system appears to be infinite, we need to truncate or simplify.

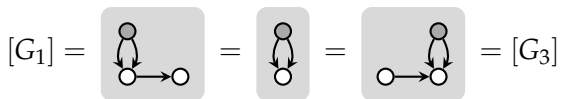
Two-legged DNA walker (cont.)

Approximation: assume infinite/circular DNA strands.



Two-legged DNA walker (cont.)

Approximation: assume infinite/circular DNA strands.

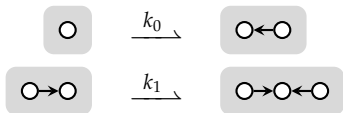


The infinite expansion reduces to a finite system of ODEs:

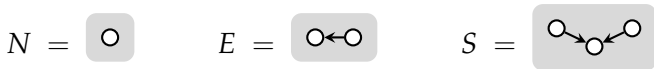
$$\begin{aligned} \frac{d}{dt} \begin{array}{c} \text{---} \circ \text{---} \\ \updownarrow \\ \circ \end{array} &= k_{F,E} \begin{array}{c} \text{---} \circ \text{---} \\ \updownarrow \\ \circ \end{array} - k_{B,C} \begin{array}{c} \text{---} \circ \text{---} \\ \updownarrow \\ \circ \end{array} - k_{F,C} \begin{array}{c} \text{---} \circ \text{---} \\ \updownarrow \\ \circ \end{array} + k_{B,E} \begin{array}{c} \text{---} \circ \text{---} \\ \updownarrow \\ \circ \end{array} \\ \frac{d}{dt} \begin{array}{c} \text{---} \circ \text{---} \\ \updownarrow \\ \circ \end{array} &= -k_{F,E} \begin{array}{c} \text{---} \circ \text{---} \\ \updownarrow \\ \circ \end{array} + k_{B,C} \begin{array}{c} \text{---} \circ \text{---} \\ \updownarrow \\ \circ \end{array} + k_{F,C} \begin{array}{c} \text{---} \circ \text{---} \\ \updownarrow \\ \circ \end{array} - k_{B,E} \begin{array}{c} \text{---} \circ \text{---} \\ \updownarrow \\ \circ \end{array} \end{aligned}$$

Example: preferential attachment

Two rules: birth and preferential attachment.²



Example observables: cells, parent-child relations, siblings.



²Example from [Danos et al., 2015]

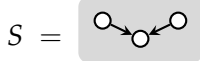
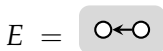
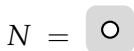
Preferential attachment (cont.)

Mean evolution of the observables.

$$\frac{d}{dt} \mathbb{E}(S) = 2(k_0 + k_1) \mathbb{E}(E) + 2k_1 \mathbb{E}(S)$$

$$\frac{d}{dt} \mathbb{E}(N) = k_0 \mathbb{E}(N) + k_1 \mathbb{E}(E)$$

$$\frac{d}{dt} \mathbb{E}(E) = k_0 \mathbb{E}(N) + k_1 \mathbb{E}(E)$$



Preferential attachment (cont.)

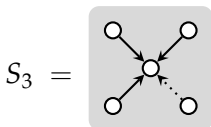
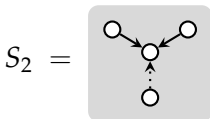
A more interesting class of observables: **degree-limited** subgraphs.

- N_k : count single nodes with exactly k neighbors (not matched).
- $[S_k]$: counts “stars”, i.e. a hub node surrounded by k neighbors (matched).

Preferential attachment (cont.)

A more interesting class of observables: **degree-limited** subgraphs.

- N_k : count single nodes with exactly k neighbors (not matched).
- $[S_k]$: counts “stars”, i.e. a hub node surrounded by k neighbors (matched).



$$[S_k] = k!N_k$$

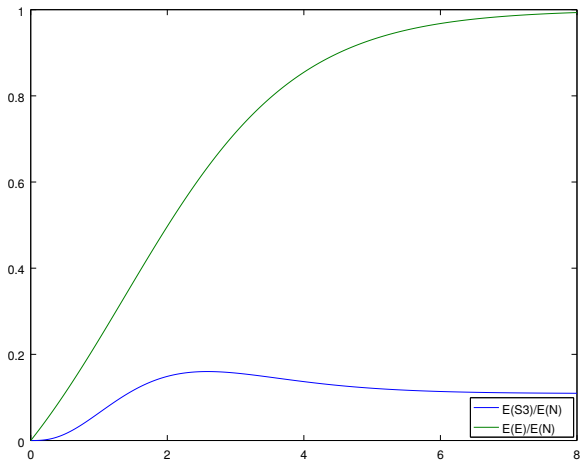
Preferential attachment (cont.)

A more interesting class of observables: **degree-limited** subgraphs.

- N_k : count single nodes with exactly k neighbors (not matched).
- $[S_k]$: counts “stars”, i.e. a hub node surrounded by k neighbors (matched).

$$\frac{d}{dt} \mathbb{E}(N_i) = (k_0 + k_1(i-1)) \mathbb{E}(N_{i-1}) - (k_0 + k_1 i) \mathbb{E}(N_i) \quad \text{for } i \geq 1$$
$$\frac{d}{dt} \mathbb{E}(N_0) = k_0 \mathbb{E}(N) + k_1 \mathbb{E}(E) - k_0 \mathbb{E}(N_0)$$

Fraction of edges and indegree-3 vertices



$$k_0 = 0.2, \quad k_1 = 0.6$$

Higher-order moments

$$\mathbb{V}_p(N_i) = \mathbb{E}_p((N_i - \mathbb{E}_p(N_i))^2) = \mathbb{E}_p((N_i)^2) - \mathbb{E}_p(N_i)^2.$$

- 11 fragments to express $[S_3]^2$,
- a total of 2097 equations to track $\mathbb{E}([S_3]^3)$,
- approx. half a minute to generate the equations,
- approx. 33 minutes to solve them using GNU/Octave.³

Tools needed...

<http://github.com/sstucki/pa-ode-gen/> generator for PA

<http://github.com/rhz/graph-rewriting/> generic tool

<http://rhz.github.io/fragger/> Java Script demo

³On a Intel Core i7 CPU.

-  Blinov, M. L., Faeder, J. R., Goldstein, B., and Hlavacek, W. S. (2004).


Bionetgen: software for rule-based modeling of signal transduction based on the interactions of molecular domains.

Bioinformatics, 20(17):3289–3291.

-  Danos, V., Feret, J., Fontana, W., Harmer, R., and Krivine, J. (2010).

Abstracting the differential semantics of rule-based models: Exact and automated model reduction.

In *LICS*, pages 362–381. IEEE Computer Society.

-  Danos, V., Feret, J., Fontana, W., and Krivine, J. (2007). Scalable simulation of cellular signaling networks, invited paper.

In Shao, Z., editor, *Proceedings of the Fifth Asian Symposium on Programming Systems, APLAS'2007, Singapore*, volume 4807 of *Lecture Notes in Computer*

Science, pages 139–157, Singapore. Springer, Berlin, Germany.



Danos, V., Heindel, T., Honorato-Zimmer, R., and Stucki, S. (2015).

Moment semantics for reversible rule-based systems.
In Krivine, J. and Stefani, J.-B., editors, *Reversible Computation – 7th International Conference, RC 2015. Proceedings*, volume 9138 of *Lecture Notes in Computer Science*, pages 3–26. Springer International Publishing.



Deeds, E. J., Bachman, J. A., and Fontana, W. (2012).
Optimizing ring assembly reveals the strength of weak interactions.

Proceedings of the National Academy of Sciences.



Feret, J., Henzinger, T., Koepl, H., and Petrov, T. (2012).
Lumpability abstractions of rule-based systems.
Theoretical Computer Science, 431(0):137–164.

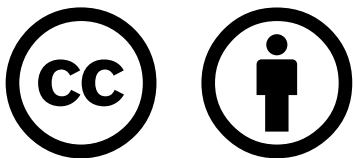
Modelling and Analysis of Biological Systems Based on papers presented at the Workshop on Membrane Computing and Bio-logically Inspired Process Calculi (MeCBIC) held in 2008 (Iasi), 2009 (Bologna) and 2010 (Jena).



Stukalin, E. B., Phillips III, H., and Kolomeisky, A. B. (2005).

Coupling of two motor proteins: a new motor can move faster.

Physical Review Letters, 94(23):238101.



Except where otherwise noted, this work is licensed under

<http://creativecommons.org/licenses/by/3.0/>